Ex.1 Working with Python Data Structures URK21CS1128

September 4, 2023

AIM: This course aims to teach learners how to work with Python data structures in data science, covering concepts, manipulation, and visualization.

DESCRIPTION: The course covers Python data structures like lists, tuples, dictionaries, sets, and arrays, along with NumPy and Pandas for data analysis. Participants will learn to visualize data and apply structures to real-world data science problems.

1. Create an empty dictionary. Fill the dictionary with prod_code and prod_name as pair by user input (Use prod_code as a key). Take one prod_code as input from the user and traverse through dictionary to find the corresponding prod_name and display the same.

[1]:
```
#1128
empty_dict = {}

n = int(input("Enter the number of product pairs you want to add: "))
for _ in range(n):
    prod_code = input("Enter the product code: ")
    prod_name = input("Enter the product name: ")
    empty_dict[prod_code] = prod_name

user_input_code = input("Enter the product code to find the corresponding␣
 ↪product name: ")
if user_input_code in empty_dict:
    print("Corresponding product name:", empty_dict[user_input_code])
else:
    print("Product code not found in the dictionary.")
```

```
Enter the number of product pairs you want to add:  2
Enter the product code:  project
Enter the product name:  book
Enter the product code:  session
Enter the product name:  guide
Enter the product code to find the corresponding product name:  session

Corresponding product name: guide
```

2. Create an empty list. Fill the list with strings by getting user input. Find the list of words that are longer than n from a given list of words. Sample List : [‘the’,’quick’,’brown’,’fox’] n : 3

[2]:
```
#1128
empty_list = []

n = int(input("Enter the number of words you want to add to the list: "))
for _ in range(n):
    word = input("Enter a word: ")
    empty_list.append(word)

min_length = int(input("Enter the minimum word length to filter the words: "))
result_list = [word for word in empty_list if len(word) > min_length]
print("Words longer than", min_length, ":", result_list)
```

```
Enter the number of words you want to add to the list:  3
Enter a word:  is
Enter a word:  was
Enter a word:  the
Enter the minimum word length to filter the words:  2

Words longer than 2 : ['was', 'the']
```

3. Create an empty set. Fill the set with values by getting user input. Check if a given value is present in a set or not.

[3]:
```
#1128
empty_set = set()

n = int(input("Enter the number of values you want to add to the set: "))
for _ in range(n):
    value = input("Enter a value: ")
    empty_set.add(value)

search_value = input("Enter the value to check if it's present in the set: ")
if search_value in empty_set:
    print(search_value, "is present in the set.")
else:
    print(search_value, "is not present in the set.")
```

```
Enter the number of values you want to add to the set:  3
Enter a value:  2
Enter a value:  3
Enter a value:  1
Enter the value to check if it's present in the set:  2

2 is present in the set.
```

4. Create an empty tuple. Populate the tuple with values by getting user input. Count the occurrence of a given input number in the tuple. Sample : (50, 10, 60, 70, 50) n : 50

[4]:
```
#1128
empty_tuple = ()
```

```
n = int(input("Enter the number of elements you want to add to the tuple: "))
for _ in range(n):
    value = int(input("Enter a number: "))
    empty_tuple += (value,)

search_number = int(input("Enter the number to count its occurrences in the␣
 ↪tuple: "))
count = empty_tuple.count(search_number)
print("Occurrences of", search_number, "in the tuple:", count)
```

```
Enter the number of elements you want to add to the tuple:  3
Enter a number:  1
Enter a number:  2
Enter a number:  3
Enter the number to count its occurrences in the tuple:  1

Occurrences of 1 in the tuple: 1
```

5. Create a 2D array and perform matrix subtraction using numpy.

[8]:
```
#1128
import numpy as np
array1 = np.array([[1, 2], [3, 4]])
array2 = np.array([[2, 1], [4, 3]])

result_array = array1 - array2
print("Result of matrix subtraction:")
print(result_array)
```

```
Result of matrix subtraction:
[[-1  1]
 [-1  1]]
```

6. Create a 2D array using numpy and find the maximum element in the matrix.

[9]:
```
#1128
import numpy as np
matrix = np.array([[5, 10, 15], [20, 25, 30], [35, 40, 45]])

max_element = np.max(matrix)
print("Maximum element in the matrix:", max_element)
```

```
Maximum element in the matrix: 45
```

7. Download the dataset from https://www.kaggle.com/datasets/varshamannem/toyato. Read the Toyota.csv file and display the basic details.

a. Display the top 10 rows
b. Display the last 5 rows
c. Display row and column details

d. Display size, shape, dimension and information summary

[6]:
```
#1128
import pandas as pd

# Assuming you have already downloaded and placed the Toyota.csv file in the current directory
file_path = "Toyota.csv"

# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# a. Display the top 10 rows
print("Top 10 rows:")
print(df.head(10))

# b. Display the last 5 rows
print("\nLast 5 rows:")
print(df.tail())

# c. Display row and column details
print("\nRow and Column details:")
print("Number of rows:", df.index)
print("Number of columns:", df.columns)

# d. Display size, shape, dimension, and information summary
print("\nSize of the DataFrame:", df.size)
print("Shape of the DataFrame:", df.shape)
print("Number of dimensions:", df.ndim)

# Information summary
print("\nInformation summary:")
print(df.info())
```

```
Top 10 rows:
   Unnamed: 0  Price   Age     KM FuelType    HP  MetColor  Automatic    CC  \
0           0  13500  23.0  46986   Diesel    90       1.0          0  2000
1           1  13750  23.0  72937   Diesel    90       1.0          0  2000
2           2  13950  24.0  41711   Diesel    90       NaN          0  2000
3           3  14950  26.0  48000   Diesel    90       0.0          0  2000
4           4  13750  30.0  38500   Diesel    90       0.0          0  2000
5           5  12950  32.0  61000   Diesel    90       0.0          0  2000
6           6  16900  27.0     ??   Diesel  ????       NaN          0  2000
7           7  18600  30.0  75889      NaN    90       1.0          0  2000
8           8  21500  27.0  19700   Petrol   192       0.0          0  1800
9           9  12950  23.0  71138   Diesel  ????       NaN          0  1900

   Doors  Weight
```

```
0  three    1165
1      3    1165
2      3    1165
3      3    1165
4      3    1170
5      3    1170
6      3    1245
7      3    1245
8      3    1185
9      3    1105

Last 5 rows:
      Unnamed: 0  Price   Age     KM FuelType   HP  MetColor  Automatic    CC  \
1431        1431   7500   NaN  20544   Petrol   86       1.0          0  1300
1432        1432  10845  72.0     ??   Petrol   86       0.0          0  1300
1433        1433   8500   NaN  17016   Petrol   86       0.0          0  1300
1434        1434   7250  70.0     ??      NaN   86       1.0          0  1300
1435        1435   6950  76.0      1   Petrol  110       0.0          0  1600

     Doors  Weight
1431     3    1025
1432     3    1015
1433     3    1015
1434     3    1015
1435     5    1114

Row and Column details:
Number of rows: RangeIndex(start=0, stop=1436, step=1)
Number of columns: Index(['Unnamed: 0', 'Price', 'Age', 'KM', 'FuelType', 'HP',
'MetColor',
       'Automatic', 'CC', 'Doors', 'Weight'],
      dtype='object')

Size of the DataFrame: 15796
Shape of the DataFrame: (1436, 11)
Number of dimensions: 2

Information summary:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  1436 non-null   int64
 1   Price       1436 non-null   int64
 2   Age         1336 non-null   float64
 3   KM          1436 non-null   object
 4   FuelType    1336 non-null   object
```

```
 5   HP          1436 non-null   object
 6   MetColor    1286 non-null   float64
 7   Automatic   1436 non-null   int64
 8   CC          1436 non-null   int64
 9   Doors       1436 non-null   object
 10  Weight      1436 non-null   int64
dtypes: float64(2), int64(5), object(4)
memory usage: 123.5+ KB
None
```

Result: The basic functionalities of data visualization using python were executed successfully.